Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus QRO

TC2037.601

Implementación de métodos computacionales (Gpo 601)

Actividad Integradora 5.3: Resaltador de sintaxis paralelo (evidencia de competencia)

Presentado por:

María Fernanda Moreno Gómez A01708653

Uri Jared Gopar Morales A01709413

Profesor:

Pedro Oscar Pérez Murueta

Fecha:

26 de mayo de 2023

Índice

Sobre este documento

En presente documento se plasman los tiempos de ejecución, speed up y el promedio de estos cálculos de 10 iteraciones de nuestro programa “Resaltador de sintaxis paralelo y secuencial en C++”, además de reflexiones acerca de estos datos, así como la complejidad algorítmica y una reflexión acerca de la ética en la tecnología y en la sociedad.

Cabe mencionar que este documento y este resaltador de sintaxis está basado en el resaltador de sintaxis para C# hecho en Racket que se hizo el mes pasado a partir de expresiones regulares para identificar las categorías léxicas de C#, por lo que por cualquier duda en el tema léxico, favor de consultar aquella evidencia.

Tiempos de ejecución en nuestro programa

Para poder evaluar con mayor precisión los tiempos de ejecución para ambas implementaciones, la paralela y la secuencial, se hicieron en total 10 iteraciones del programa, las cuales fueron plasmadas en la siguiente tabla, identificando el tiempo en milisegundos para la implementación secuencial, paralela y el speedup:

Num. Iteración	Paralelo	Secuencial	Speedup
1	5001.57	15958.4	3.19069
2	4578.23	17304.1	3.77963
3	4538.17	17049.7	3.75695
4	4569.07	16840.7	3.6858
5	5206.31	15257.7	2.93061
6	5178.98	17578.3	3.39416
7	4268.53	15480.2	3.62659
8	4011.24	15305.5	3.81565
9	4126.02	18034.3	4.37087
10	4003.7	15012.1	3.74956
Promedio	4548.182	16382.1	3.630051

Tabla 1.0. Tiempos de ejecución en milisegundos del resaltador de sintaxis para ambas implementaciones, así como el speedup que se tuvo con la implementación paralela.

El cálculo de Speedup se pudo obtener gracias a los tiempos de ejecución tanto para la implementación paralela como la secuencial, ya que para obtener este dato, se divide el tiempo secuencial entre el tiempo en paralelo, obteniendo de esta manera la cuarta columna.

De esta manera tenemos un tiempo promedio de ejecución de la implementación paralela de 4,548.182 ms (milisegundos) y un tiempo promedio de ejecución de la implementación secuencial de 16,382.1 ms (milisegundos), de manera que tenemos un speedup de 3.63.
Evidentemente, la versión paralela fue un montón de veces más rápida que la secuencial. En promedio, fue como 3.6 veces más rápida. Eso es mucho tiempo que podríamos ahorrar si tuviéramos que resaltar un montón de código. Sin embargo, hay áreas de mejora que podríamos trabajar en ellas para mejorar aún más el tiempo de ejecución en ambas implementaciones:

- Las expresiones regulares, muchas de ellas son palabras enteras, por lo que hay demasiadas, entonces se podría mejorar la forma de buscarlas para procesarlas de una mejor manera.
- La escritura de los archivos en HTML es secuencial, lo que lo hace lento, por lo que podríamos mejorar esta parte.
- Usar hilos que se creen y destruyan para optimizar el programa.

Entonces, en resumen, creo que hicimos un gran trabajo en este proyecto, los tiempos de ejecución no son extremadamente altos, y el tiempo de la ejecución de la implementación secuencial si fue mayor a la paralela, por lo que hicimos bien ambas implementaciones. Pero siempre hay formas de mejorar, y creo que tenemos algunas ideas realmente buenas para hacer nuestro resaltador aún mejor en un futuro.

Complejidad algorítmica

La complejidad algorítmica en el peor de los casos es **O(n^2)**, esto es debido a la complejidad de las expresiones regulares y que entre más grandes sean los archivos de entrada, va a crecer el tiempo de ejecución.
Sin embargo, al usar hilos el tiempo total de ejecución se reduce al número de hilos, pues se le da a cada uno de estos trabajo para que lo ejecute, esto se vio reflejado en el speedup de 3.6 veces más rápida la implementación de la manera paralela en comparación con la secuencial.

Reflexión final

Podemos concluir que la utilización de expresiones regulares fueron muy útiles para identificar las ocasiones donde se debían de resaltar de manera distinta cada categoría léxica de C#.

Ahora, en cuanto a la complejidad algorítmica, las cosas pueden ponerse un poco complicadas. Las expresiones regulares pueden ser un poco costosas en términos de rendimiento, debido a la diferencia entre estas, pero en nuestro caso, las expresiones regulares que usamos eran bastante sencillas y directas. Así que, en general, nuestro algoritmo tenía una complejidad más o menos lineal con respecto al tamaño del código fuente que estábamos analizando.

Lo verdaderamente sorprendente e importante ocurrió cuando utilizamos la programación paralela, pues al dividir el trabajo en hilos, pudimos acelerar significativamente nuestro resaltador. El tiempo de ejecución fue en promedio 3.6 veces más rápido con 8 hilos, lo cual es un gran logro. Aunque es importante recordar que no es una aceleración perfecta. No vimos una aceleración de exactamente 8 veces, probablemente debido al tiempo adicional requerido para la creación de hilos y la sincronización cuando escribimos los resultados en los archivos HTML.

Acerca de las implicaciones éticas que el tipo de tecnología pudiera tener en la sociedad, nos gustaría decir un par de cosas. En primer lugar, lo más importante que podemos hacer como ingenieros de ITC es asegurarnos de que las herramientas que desarrollamos se utilicen de manera responsable y ética. En nuestro caso, creamos una herramienta de resaltado de sintaxis C para C# que puede parecer bastante inofensiva a primera vista. Pero si lo piensas bien, estas herramientas son realmente poderosas. Por ejemplo, esta función de resaltado de sintaxis puede facilitar mucho la vida de los programadores al permitirles comprender y revisar el código de forma más rápida y sencilla. Esto puede conducir a un software de mayor calidad, menos errores y, en última instancia, productos más seguros y confiables para el público. Sin embargo, como cualquier herramienta, existe la posibilidad de abuso. Si se usa incorrectamente, dicha herramienta puede usarse para enriquecer o plagiar código, lo cual es claramente inaceptable.

Siempre debemos de usar la tecnología y las herramientas que creamos para hacer de nuestra vida, una vida más cómoda, no para dañarnos, sino para ayudarnos.